File: `bisection.8th`

Find a zero of the given function $f(x)$ in the interval (a, b). For best performance, we should have $f(a)f(b) < 0$, i.e. the values should have opposite signs.

Set `num-debug` to true, if you want to see the iterations.

Set `lines` to something like 5 or 10 if you have to much output.

If you need more accuracy than the machine accuracy of 15 secure places, you can use something like `30 big-floats`.

You should also consult the documentation of the numerics library.

```
needs tools
needs numerics
```

We store $f(a)$ to not have to evaluate it twice.

```
var f(a)
```

This is a test function: $f(x) = x^3 + 4x^2 - 10$. We evaluate it using Horner's Schema: $(x + 4)x^2 - 10$.

```
: f-test \ n -- n
   dup 4 n:+ swap n:sqr n:* 10 n:- ;
```

Calculate the midpoint between a and b using $p = a + \frac{b-a}{2}$, which is safer against overflow than $\frac{a+b}{2}$.

```
: midpoint \ a b -- a+(b-a)/2
   over n:- 2 n:/ n:+ ;
```

Print a row of our data. We have: n iteration counter, a left interval end, b right interval end, p midpoint, $f(p)$ value at midpoint, $|a - b|$ interval length, if $l > 1$, only every l-th line is shown. Note: `print-row` only displays anything if `num-debug` is true.

```
: print-row \ n a b p f(p) l -- n a b p f(p)
   nm:num-debug @ if
      5 pick swap n:mod 0 = if
         4 #> 4 pick . " " .
         3 pick nm:f. " " .
         2 pick nm:f. " " .
         1 pick nm:f. " " .
```

```
      0 pick nm:f. " " .
      3 pick 2 pick n:- n:abs nm:f.
      cr
   then
else drop then ;
```

Exit criterion. Returns true iff $f(p) < \epsilon$ or $|b - p| < \epsilon$.

```
: criterion \ n b p f(p) -- n b p f(p) ?
   dup 0 nm:eps n:~ >r
   -rot 2dup nm:eps n:~ >r rot
   2r> or ;
```

Takes an interval (a, b) and returns a new interval, which is either (a, p) or (p, b) depending on which the zero lies in.

```
: next-interval \ a b n --  a p | p b
   dup nm:max-iterations @ n:= if ;; then
   -rot 2dup midpoint dup nm:f \ n a b p f(p)
   criterion if
      1 print-row
      break
   else
      nm:print-lines @ print-row
      f(a) @ over n:* 0 n:> if \ n a b p f(p) f(a)*f(p)>0
         f(a) ! rot drop swap \ n a b p
      else
         drop nip \ n a b
   then then rot drop ;
```

Given an initial interval, that includes at least one zero, `bisection` returns an interval with a zero in it.

```
: bisection \ a b -- a b
   over nm:f f(a) !
   nm:num-debug @ if
      "n     a                    b                    p                 "
      "      f(p)                 |a-b| " s:+ . cr
      124 draw-line cr
   then
   ' next-interval 0 nm:max-iterations @ loop
   nm:num-debug @ if 124 draw-line cr then
   nm:max-iterations @ n:= if
      "ran " . nm:max-iterations @ .
      " iterations, but the result was not close enough" . cr
   then ;
```

Show the steps

```
true nm:num-debug !

' f-test w:is nm:f
1. 2. bisection .s 2dup nm:f. nm:f. n:- n:abs nm:f. cr

bye
```